

SemiFed: Semi-supervised Federated Learning with Consistency and Pseudo-Labeling

Haowen Lin¹, Jian Lou^{2,3,*}, Li Xiong², Cyrus Shahabi¹

¹University of Southern California ²Emory University ³Xidian University
haowenli@usc.edu jlou@xidian.edu.cn lxiong@emory.edu shahabi@usc.edu

August 24, 2021

Abstract

Federated learning enables multiple clients, such as mobile phones and organizations, to collaboratively learn a shared model for prediction while protecting local data privacy. However, most recent research and applications of federated learning assume that all clients have fully labeled data, which is impractical in real-world settings. In this work, we focus on a new scenario for cross-silo federated learning, where data samples of each client are partially labeled. We borrow ideas from semi-supervised learning methods where a large amount of unlabeled data is utilized to improve the model's accuracy despite limited access to labeled examples. We propose a new framework dubbed SemiFed that unifies two dominant approaches for semi-supervised learning: consistency regularization and pseudo-labeling. SemiFed first applies advanced data augmentation techniques to enforce consistency regularization and then generates pseudo-labels using the model's predictions during training. SemiFed takes advantage of the federation so that for a given image, the pseudo-label holds only if multiple models from different clients produce a high-confidence prediction and agree on the same label. Extensive experiments on two image benchmarks demonstrate the effectiveness of our approach under both homogeneous and heterogeneous data distribution settings.

Keywords— federated learning, semi-supervised learning

1 Introduction

While deep neural networks have shown promising results in various computer vision applications and tasks, their success is largely attributed to the availability of a significant amount of training data [6, 17, 19]. Fortunately, with the rapid proliferation of smart devices and sensors, multiple clients such as mobile device users or organizations can collaboratively learn a global model

via communication, thus alleviating the burden on data gathering for a single entity. Traditional machine learning algorithms require centralizing all data samples in a single machine or data center, which is impractical due to privacy concerns, high storage costs, and legal constraints. This motivates federated learning (FL) [20], which is a promising machine learning paradigm in which a loose federation of clients participate in collaborative training under the coordination of a central server.

Although federated learning has attracted much attention in both academia and industry, existing FL applications and approaches mainly focus on fully supervised settings where all the input data have labels. However, this assumption is not practical, because manually adding high-quality labels to all training data may not be equally feasible by all clients. A relevant approach in centralized learning is semi-supervised learning (SSL), which leverages unlabeled samples in addition to a small portion of labeled examples to obtain performance gains. However, it is not trivial to marry centralized SSL techniques with FL training due to its distinctive requirements, rendering a direct application of SSL impractical for real FL settings.

One of the most challenging problems in FL setting is that data among clients can be non-independent and identically (non-IID). In this case, even labeled and unlabeled data may come from different underlying distributions, and the unlabeled data may contain classes not present in the labeled data. This non-IID situation can have a paramount impact on the SSL algorithm design. For example, [24] shows that even in centralized setting, utilizing data from a mismatched set of classes can even hurt the performance of many SSL methods compared to not using any unlabeled data at all. While [24] synthetically varies the class overlap, this mismatch between labeled and unlabeled categories naturally arises under the non-IID data partitions in FL setting. Therefore, it is important to design a method

*Corresponding Author.

that is not sensitive to the non-IID data distributions and does not deteriorate by adding unlabeled data from a mismatched set of classes.

We propose Semi-Supervised Federated Learning (SemiFed) as a unified framework and apply it to image classification with limited labeled samples. Figure 1 presents the illustration of the limited labeled data under collaborative training scenario. SemiFed consists of two key components. First, it performs consistency regularization, which encourages the network to produce the same output distribution when its inputs are perturbed and can be applied to all samples without labels. We carefully choose the noise injected into consistency training. We find that advanced data augmentation [7, 8] techniques that boost network accuracy in fully supervised tasks can be leveraged for the perturbations in our FL semi-supervised learning. Second, after several rounds of warm-up training phases, we adopt the idea of pseudo-labeling [16] which first produces an artificial label for each unlabeled image and then enforces the model to predict the artificial label when fed the unlabeled sample as input in the following training stages. However, training the network with falsely inferred pseudo-labels may degrade the model performance. To solve this issue, we keep the artificial labels only when the local models and the global model assign a very high probability (high confidence) to one of the possible classes. Further, to take advantage of the federation, we transmit local models and assign a pseudo-label to an unlabeled sample only when multiple models agree on the same label candidate with high confidence. With the integration of consistency regularization and federated pseudo-labeling, we conduct experiments to evaluate the proposed approach on standard image benchmarks under federated learning setting. We show that our algorithm is consistent under non-IID distribution data, which is significant under FL setting.

The remainder of the paper is organized as follows. Section 2 presents preliminaries of FL and related work. Section 3 describes the scenario and methodology of SemiFed. Section 4 presents the experiments and evaluation of the results. Finally, Section 5 concludes the paper with a discussion of future work.

2 Background and Related Work

In this section, we first formally define the problem of SSL in federated learning where clients may only create or obtain a small portion of labels for their local datasets. Then, we present related work that attempts to solve the problem of limited labeled samples in federated learning.

1) *Problem setting.* Let \mathcal{D} be the dataset with C

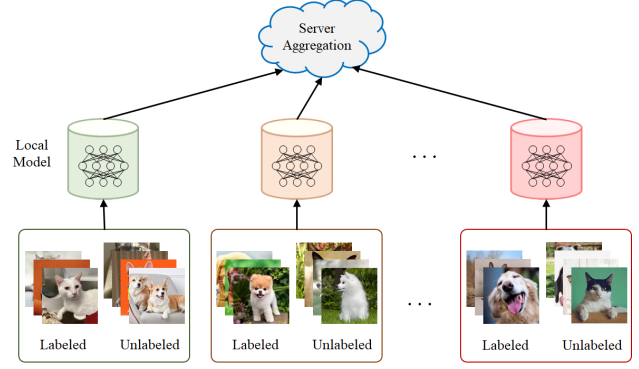


Figure 1: Illustration of a realistic Federated Semi-supervised learning scenario

categories. We assume that there are K clients (e.g., edge devices) participating in the federated learning. Each client has a dataset \mathcal{D}^k with C categories that are separated into labeled dataset $\mathcal{D}_l^k := \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{N_l^k}$ with \mathbf{x}_i denoting the i -th sample and $\mathbf{y}_i \in \{1, 2, \dots, C\}$ being the corresponding label, and unlabeled dataset $\mathcal{D}_u^k := \{\mathbf{x}_j\}_{j=1}^{N_u^k}$. The total number of data samples across all clients is therefore $N_t = \sum_{i=1}^K (N_l^i + N_u^i)$. N_l^k is the number of labeled data samples for k th client, N_u^k is the number of unlabeled data samples, and usually, the labeled data are much less than unlabeled data, i.e., $N_l^k \ll N_u^k$. Depending on the data distribution, the local datasets \mathcal{D}^k could either be homogeneous, i.e. independent and identically distributed (IID)), or heterogeneous with varying distribution from clients to clients (i.e., non-IID).

2) *Related work.* Although limited, there still exist some attempts to the limited labeled samples problem in federated learning. Guha et al. [10] propose one-shot federated learning, where they let the server learn a successful global model over a network of federated devices in a single round of communication in both supervised and semi-supervised learning. In their setting, all clients have labeled data, and only the central server has unlabeled data. Sharma et al. propose to solve federated transfer learning under Secret Sharing protocols. They assume that each client can have different datasets with totally different labels. However, we focus on the scenario when the server is not allowed to have any data for privacy reasons, which is different from the setting introduced in previous work. Lastly, a parallel work that targets this challenge is [14], where it imposes consistency loss among multiple clients and decomposes model parameters for labeled and unlabeled data to exploit unlabeled information. However, they do not provide detailed experiment results to show model

performance under the different number of labeled samples under IID and non-IID settings.

3 SemiFed Framework

SemiFed unifies two powerful approaches that can leverage the unlabeled data at each local site. First, SemiFed utilizes consistency regularization to leverage unlabeled data. Second, it uses pseudo-labeling to progressively label unlabeled data using the models trained so far to guide the remaining learning process. As a key innovation, we also adapt the pseudo-labeling method to FL setting, where we require multiple local models and the global model to agree on the labels before we make use of the labels for future training.

3.1 Consistency Regularization Consistency regularization is one of the most significant approaches to learn from unlabeled data and improve the performance of labeled data [2, 21]. In a nutshell, Consistency Regularization utilizes the unlabeled data by enforcing model predictions to be invariant under any kind of small random transformation of the data and perturbations to the model. The design relies on the low-density separation assumption that the decision boundary should not traverse samples of high-density regions but should stay in low-density regions [32]. Thus, the model should be robust to any small changes fed into the input or any hidden layer.

Following the work in [2, 21], we formulate the objective function to incorporate information from both labeled and unlabeled data at each client. It includes two loss terms: a supervised loss l_s^k applied to labeled data and an unsupervised loss l_u^k applied to data without class label at k th client. Specifically, we define our l_s as the standard cross-entropy loss on the labeled examples where p_w^k is the local model in k -th client:

$$(3.1) \quad l_s^k = \sum_{i=1}^{N_l^k} -\log p_{\omega}^k(\mathbf{y}_i | \mathbf{x}_i).$$

For unlabeled data, we focus on the setting where the noise is fed into the input \mathbf{x} (i.e. $\tilde{\mathbf{x}} = a(\mathbf{x}, \epsilon)$). We represent the distribution of the current predictions of samples $p_{\omega}(\mathbf{y} | \mathbf{x})$ and generate an auxiliary distribution $p_w(\mathbf{y} | \tilde{\mathbf{x}})$ from the perturbed data $\tilde{\mathbf{x}}$. Our goal is to minimize the distance between these two distributions using the KL divergence, a widely used distribution-wise asymmetric distance measure. Therefore, the unsupervised loss for k th client is defined as:

$$(3.2) \quad l_u^k = \text{KL}(p_{\omega}^k(\mathbf{y} | \mathbf{x}) || p_w^k(\mathbf{y} | \tilde{\mathbf{x}})).$$

For jointly training with all data samples, the full

objective function is thus given by

$$(3.3) \quad l_s^k + \lambda_u l_u^k,$$

where λ_u is a hyper-parameter that controls the balance between the supervised cross entropy and the unsupervised consistency training loss.

Previous works have shown that one effective way to improve the performance of this consistency training procedure is to “wisely” choose the type of noise injection [29]. A straightforward way is to employ random noise functions such as Gaussian noise and dropout noise in the vision domain [15]. However, simply injecting a random noise cannot effectively push the boundary line to cross the low-density region or can make the predictor unstable under a small perturbation in a specific direction [9]. To alleviate this problem, [21] searches adversarial perturbations that maximize the change in model prediction. [3, 4] generates noise by interpolating random unlabeled data points. More recent work shows that using stronger data augmentations such as augmentation policies searched by reinforcement learning not only improves the generalization of deep learning models but also can be beneficial to the objective in consistency learning framework [7].

Following the idea in [30], we take advantage of the advanced data augmentation technique called RandAugment [8] that is initially inspired by AutoAugment [7]. AutoAugment uses a search method to automatically find effective data augmentation strategies that include all transform operations in the Python Image Library (PIL) for a specific dataset. RandAugment does not require a separate search phase on a proxy task but instead uniformly takes operation from the PIL library with similar performance compared with AutoAugment. Because we do not have enough labels for the dataset in our settings, we explore RandAugment that does not require any labeled data to search for the policy. We choose such advanced data augmentation as our noise injection method because it can generate meaningful augmented samples under large modifications to the data input. We show that using RandAugment on the training set yields positive results in the experiment.

3.2 Pseudo-Labeling Our algorithm also incorporates the idea of pseudo-labeling [16], a simple but efficient method that can further improve the performance of classifier when we train using the labeled and unlabeled data simultaneously. Pseudo-labeling first trains the model only on labeled dataset. At each iteration, the model uses predictions from previous iteration as target classes for unlabeled data samples as if they were true labels. The process goes through fixed number of

iterations until all unlabeled data are labeled. Pseudo-label advances the self-training in the way that it would use both labeled data and unlabeled data for training [16]. For each unlabeled data sample, it would pick the class with maximum predicted confidence as prediction to be used as a pseudo label,

$$(3.4) \quad \tilde{y}_c = \begin{cases} 1, & \text{if } c = \operatorname{argmax}_c p_{\omega}(\mathbf{x})[c] \\ 0, & \text{otherwise,} \end{cases}$$

where \tilde{y}_c is the pseudo label and $p_{\omega}(\mathbf{x})[c]$ is the model prediction value of sample \mathbf{x} for class c .

Algorithm 1 Pseudo-Labeling Procedure

Input: \mathcal{D}_l^k set of labeled samples in client k , set of unlabeled samples \mathcal{D}_u^k , model agreement number u , dictionary P that stored all model parameters
for each client $i \in S$ **in parallel do**

```

    U :=  $\mathcal{D}_l^k$ 
    L :=  $\mathcal{D}_l^k$ 
    for  $\mathbf{x} \in U$  do
        for each model  $p_{\omega}^k$  do
            |  $\tilde{y}^k \leftarrow$  using equation (3.5) to predict on  $\mathbf{x}$ 
        end
         $s_x \leftarrow$  the number of most frequent element in local model predictions
        if  $s_x \geq u$  then
            | U :=  $\mathcal{D}_l^k \cup (\mathbf{x}, \tilde{y}_i)$ , where  $\tilde{y}_i$  is defined by equation (3.6)
            | L :=  $\mathcal{D}_l^k \setminus (\mathbf{x}, y)$ 
        end
    end
end

```

In our setting, each client in communication round t has labeled data $\mathcal{D}_{l_t}^k := \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N_{l_t}}$, unlabeled data $\mathcal{D}_{u_t}^k := \{\mathbf{x}_j\}_{j=1}^{N_{u_t}}$ and local model p_{ω}^k , $t \in 1, \dots, T$. When the subset of unlabeled data $\tilde{\mathcal{D}}_{u_t}^k$ satisfies a predefined criterion, as shown in the following paragraph, the model will generate pseudo label as shown in equation 3.4. Thus, the new labeled data and unlabeled data are generated as $\mathcal{D}_{l_{t+1}}^k = \mathcal{D}_{l_t}^k \cup \tilde{\mathcal{D}}_{u_t}^k$, $\mathcal{D}_{u_{t+1}}^k = \mathcal{D}_{u_t}^k \setminus \tilde{\mathcal{D}}_{u_t}^k$. In this way, the model propagates hard pseudo-labels (i.e. one-hot vectors) to the unlabeled data. We train the local model p_{ω}^k employing standard cross entropy loss to both annotated true labels and pseudo-labels equally.

The criteria used to select how many and which samples are transferred from unlabeled data to the labeled data during training at each round is key to our method. Model predictions that generated these hard labels are not all correct. It is important to reduce the error rate of pseudo-labels so that the error

does not get propagated into future training. Previous studies have explored different types of uncertainty of the prediction to alleviate this problem, including using the highest confidence to label data [32], or performing label propagation based on the nearest feature space [27]. We adopt the former approach but additionally use multiple clients to help with pseudo-label in our federated learning setting. At communication round t , the server broadcasts not only the aggregated model but also K other local models to each client. Thus, each client would have $K+1$ models stored locally and generate pseudo-labels based on multiple model agreements. Consider \mathbf{x}_i as an unlabeled data point. \mathbf{x}_i is fed to these $K+1$ models in parallel, where we can get $\tilde{\mathbf{y}}_i^k = \mathbf{1}(p_{\omega}^k(\mathbf{x}))$ with $\mathbf{1}(\cdot)$ producing one-hot labels with given softmax values. To maintain stability, in each client, we only retain artificial labels whose largest class probability is above a predefined threshold γ_t for communication round t [16].

$$(3.5) \quad \tilde{\mathbf{y}}_i^k = \mathbf{1}(p_{\omega}^k(\mathbf{x}) \geq \gamma_t).$$

In addition, we incorporate an unlabeled data point based on how many models agree on the same label. To generate a pseudo-label for data sample, we have

$$(3.6) \quad \tilde{\mathbf{y}}_i = \text{Mode}(\mathbf{y}_i^1, \mathbf{y}_i^2, \dots, \mathbf{y}_i^{K+1}),$$

where $\text{Mode}(\cdot)$ outputs the class category that has maximum model agreement. We then only keep the label if enough local models agree on the sample label (i.e. using a predefined model agreement number u , $u \leq K+1$). Algorithm 1 shows the pseudo-labeling procedure. The output of the algorithm is the updated local labeled and unlabeled datasets. The entire training procedure is presented in Algorithm 2.

4 Experimental Evaluation

4.1 Experiment Setup In this section, we quantitatively evaluate our algorithm with extensive experiments on two image classification benchmarks.

1) *Dataset.* We evaluate our framework on two standard image classification datasets: CIFAR-10 [23] and Street View House Numbers (SVHN) [22]. Both datasets have resolution 32×32 . In each dataset, we keep a small portion of the training images labeled and leave the rest of the dataset unlabeled. We evaluate the performance on the independent test set. The details are as follows.

CIFAR-10 contains 10 classes with 50K color images for training and 10K for testing. As recommended by [24], we use a validation set of 5K samples for CIFAR-10 for choosing hyper-parameters and add it back to the



Client ID	Numbers of Samples in the Classes										Distribution
	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	
k=0	5	35	231	3	75	3	2	6	17	42	
k=1	22	15	0	147	6	11	3	22	53	28	
k=2	11	2	33	109	2	0	39	83	96	0	
k=3	69	3	4	27	11	117	34	6	35	1	
k=4	24	4	0	14	205	4	21	73	1	62	
k=5	26	24	50	79	28	40	8	48	16	266	
k=6	0	57	5	2	41	72	66	59	111	0	
k=7	182	138	45	10	30	0	0	0	0	0	
k=8	59	68	10	3	0	67	183	1	20	0	
k=9	2	54	22	6	2	86	44	102	51	1	

Client ID	Numbers of Samples in the Classes										Distribution
	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	
k=0	91	40	237	659	37	599	1043	2	988	41	
k=1	1323	6	109	21	372	254	41	1812	110	702	
k=2	788	9	2401	0	58	43	130	619	538	22	
k=3	41	401	382	11	2020	1354	106	38	27	124	
k=4	1326	2	294	10	326	1963	879	0	0	0	
k=5	26	0	131	146	297	111	1836	432	46	1445	
k=6	79	506	630	0	495	134	203	1541	352	915	
k=7	4	83	19	155	413	114	361	9	2509	1278	
k=8	508	62	325	1119	582	28	1	147	30	73	
k=9	414	3491	72	2479	0	0	0	0	0	0	

Figure 2: The distribution of non-IID data for CIFAR-10. The left table shows the data distribution for 4000 labeled images and the right table shows the data distribution for 46000 labeled images

Table 1: Accuracy of federated learning with varying number of labeled sample on CIFAR-10 and SVHN using ResNet-50 architecture under IID data partition settings.


Method	CIFAR-10			SVHN		
	1000 labels	2000 labels	4000 labels	1000 labels	2000 labels	4000 labels
Supervised	46.97	60.31	66.57	49.17	82.93	86.43
VAT	50.82	67.10	80.04	88.70	92.48	93.79
UDA-consistency	50.13	68.25	77.68	85.35	92.45	94.80
SemiFed	52.63	72.98	81.34	89.50	94.43	94.45
Fully-Supervised	91.03			94.81		

training set to evaluate test accuracy. We report results using 1K, 2K, and 4K labeled samples.

SVHN has 73257 images for training and 26032 for testing. Similarly, we perform experiments with a varying number of labeled images $N_l = 1K, 2K, 4K$. The test images are used for a global test after each round.

2) *Non-IID data preparation.* In practical federated learning settings, we often encounter non-IID data at different clients. In the experiments, we study the case when different clients i and j have different data distribution P_i and P_j , which follows the practice of many prior works [13, 20]. We report results on both IID and non-IID settings. We generate non-IID data partitions of each dataset by splitting training samples into 10 unbalanced partitions. Following the scheme in [11, 31], we generate the non-IID portion by sampling $p_c \sim \text{Dir}_J(0.5)$, which is Dirichlet distribution with concentration parameter 0.5 and allocating a $p_{c,k}$ proportion of the training samples of class c to local client k . The actual heterogeneous data distribution is presented in Figure 2. Note that we do not use currently released federated learning benchmark datasets such as FEMNIST because they are too “easy” to classify, i.e., the model trained on a small subset of labeled data can already achieve high performance.

3) *Implementation details.* Unless otherwise specified, we use the following experiment implementation as the standard setting: we setup 10 clients and run on a GPU server for all datasets and models. We test our framework using ResNet-50 [12] as the base model architecture for the CIFAR-10 and SVHN datasets. There are several important hyper-parameters in our SemiFed framework: the number of total communication rounds, the edge-side epoch number, the unsupervised balancing parameter λ_u , the predefined confidence threshold γ_t and the model agreement number u . After a grid search for tuning parameters, we run all experiments with 300 communication rounds, unsupervised ratio $\lambda_u = 1.0$, model agreement number $u = 11$ (all local models and the global model have to agree on the same label for a single data point if we want to pseudo-label it). We will also show a detailed parameter study in Section 4.3. We apply pseudo-labeling method at communication round $t = 50, 100$ for SVHN and $t = 50, 100, 200$ for CIFAR-10. For the confidence threshold, we set 0.98 for SVHN and IID data partitions of CIFAR-10 and 0.9, 0.85, 0.7 for non-IID partitions of CIFAR-10. We choose the confidence threshold based on how many samples are given labels (up to 1000 per round). The client-side epoch number depends on the dataset. For CIFAR-10, we set it to be 10, and for SVHN we set it to be 5 since SVHN

Table 2:  Accuracy of federated learning with varying the number of labeled samples on CIFAR-10 and SVHN using ResNet-50 architecture under non-IID data distribution settings.

Method	CIFAR-10			SVHN		
	1000 labels	2000 labels	4000 labels	1000 labels	2000 labels	4000 labels
Supervised	40.26	56.87	61.31	50.62	80.00	84.82
VAT	45.19	60.26	67.85	83.76	90.27	93.14
UDA-consistency	44.29	58.24	71.57	87.88	91.42	93.30
SemiFed	46.46	62.63	75.41	89.50	92.81	94.01
Fully-Supervised	88.90			92.99		

Algorithm 2 SemiFed, $|\mathcal{B}_l|$, $|\mathcal{B}_u|$ is the local mini-batch size for labeled and unlabeled data, T is the number of communication rounds, and η is the learning rate, E is the number of local epochs, I is the number of local iterations within each epoch, λ_u is the coefficient for unsupervised loss, T_p is the set of communication rounds that perform pseudo-labeling

Server executes:

Initialize ω_0 ;

for each round $t = 0, \dots, T - 1$ **do**

$\omega_{t,0}^i = \omega_t$;

$P \leftarrow$ empty dictionary

for each client $k \in \mathcal{S}$ **in parallel** **do**

$\omega_t^k \leftarrow \text{ClientLocalTraining}(k)$

$P[k] = \omega_t^k$

end

$\omega_{t+1} = \frac{1}{|\mathcal{S}|} \sum_{k \in \mathcal{S}} \omega_t^k$;

for each client $k \in \mathcal{S}$ **in parallel** **do**

if $t \in T_p$ **then**

 send model dictionary P to client k

end

 send aggregated model ω_{t+1} to client k

end

end

ClientLocalTraining(k):

for local epoch e from 0 to $E - 1$ **do**

for iteration i from 0 to $I - 1$ **do**

 Sample mini-batch \mathcal{B}_l from \mathcal{D}_l^k ;

 Sample mini-batch \mathcal{B}_u from \mathcal{D}_u^k ;

 // l_s is computed using Eq. 3.1

 // l_u is computed using Eq. 3.2

$\omega_{t,i+1}^k = \omega_{t,i}^k - \eta \cdot \frac{1}{|\mathcal{B}_l|} \sum_{b_l \in \mathcal{B}_l} \nabla l_s(\omega_{t,i}^k; \mathbf{x}_{l,b}, \mathbf{y}_b) -$

$\lambda_u \cdot \eta \cdot \frac{1}{|\mathcal{B}_u|} \sum_{b_u \in \mathcal{B}_u} \nabla l_u(\omega_{t,i}^k; \mathbf{x}_{u,b})$

end

if $t \in T_p$ **then**

 Execute Algorithm 1

end

end

is much easier to classify. On the client-side, we use Stochastic Gradient Descent with nesterov momentum as the optimizer, where the momentum factor is 0.9, L_2 regularization is 0.0001, the constant learning rate is $lr = 0.3$, and the batch size is 64.

Current popular distributed training libraries in PyTorch [25] and TensorFlow [1] (i.e. TensorFlow-Federated (TFF), PySyft [26], and LEAF [5]) lack the support of our algorithm (pseudo-labeling with multiple model transferring). To simplify the usage of our algorithm, we build our algorithm on FedML [11], an open-source library for federated learning and deploy it in a distributed computing environment. We conduct the experiments on a computer with three NVIDIA GeForce GTX 2080 GPUs to report the performance.

4) *Compared Baselines.* We evaluate the following approaches:

- **Supervised.** Supervised refers to using only labeled data points from CIFAR-10 and SVHN respectively, without any unlabeled data on the client-side. The server still performs FedAvg and aggregates local models and broadcasts the aggregated global model to all clients.
- **Fully Supervised.** we also train a fully supervised baseline (use all the original labels of the training samples) to measure the the ideal accuracy we could hope to obtain under federated learning settings.
- **VAT.** VAT stands for Virtual Adversarial Training [21]. It applies a small perturbation to both the labeled and unlabeled input in the direction that can mostly alter the prediction of the model from the current inferred label by the model.
- **UDA-consistency [30].** On the client-side, UDA uses consistency loss only to enforce the model being insensitive to the noise added to the input and hence smoother with respect to changes in the input space. It relies on advanced data augmentation [8] which is the same augmentation technique

in our consistency regularization to add the noise. We re-implement this method in federated learning to give a fair comparison.

4.2 Performance Comparison with Baselines

We evaluate the algorithm on CIFAR-10 and SVHN datasets using ResNet-50 and report our results under IID and non-IID setting in Table 1 and Table 2, respectively. The relative performance against the baseline approaches vary across different datasets with a different number of labeled samples. Clearly, our algorithm outperforms the supervised method that trains the network using only labeled samples by a large margin. We find that even using consistency loss alone can improve model performance a lot compared with the model trained with only labeled data, demonstrating the advantage of consistency-regularization based method exploiting unlabeled information. Note that under SVHN non-IID data distribution with our consistency regularization and pseudo-labeling method, we achieve accuracy better than using all labeled data without advanced data augmentation. Overall, our approach improves the test accuracy with an improvement ranging from 0.71% to 4.39% improvement on CIFAR-10 and matches the test accuracy on SVHN when compared with all previous methods that only rely on consistency loss [30].

4.3 Parameter Study Deep neural networks trained for image classification can be heavily affected by the architecture, training schedule, etc. Moreover, since our algorithm is built upon the assumption where decision boundary should lie in low-density regions to improve generalization performance, other factors such as the L_p regularization term should be significant. Thus, we want to quantitatively show the importance of these factors and understand the effects of different parameter settings of our algorithm on the final results. We focus on studying with a single 4000 label split from CIFAR-10 under non-IID data partitions and report results using constant learning rate $lr = 0.3$ for consistency regularization.

1) *Effect of the unsupervised ratio λ_u .* Parameter λ_u controls the weight balance between the labeled data and the unlabeled data loss. λ_u being too small would cause the model to be mainly controlled by supervised loss and to learn nothing from unlabeled data. On the other hand, if λ_u is too large, the model would neglect useful guidance from labels. We illustrate the effect of varying unsupervised ratio on the validation performance in Figure 3a. We found that both $\lambda_u = 1.0$ and $\lambda_u = 2.0$ yields good performance. Following the setting in [21], we choose $\lambda_u = 1.0$ in all our experiment.

2) *Effect of the L_2 regularization in client-side.* We

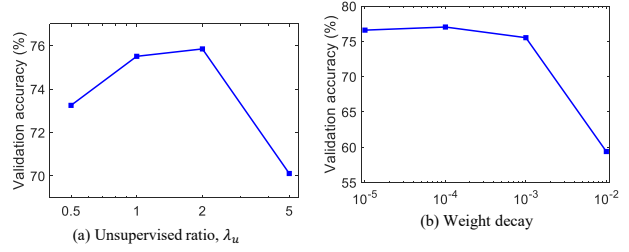


Figure 3: Plots of various parameter studies on SemiFed (a) Varying the ratio of unsupervised loss (b) Varying the loss coefficient for weight decay.

apply L_2 regularization in all our experiment. L_2 regularization is particularly important since it can have an effect of mitigating the over-sensitivity of the output with respect to input. Figure 3b. demonstrates the effect of choosing the loss coefficient for L_2 regularization. We find that choosing a large coefficient can degrade performance by five percent.

3) *Effect of the learning rate in client-side.* In Table 3, we compare the training curves with different learning rates and learning rate decay schedules. We find that the learning rate can have a substantial effect on network performance. Given an inappropriate learning rate, i.e., $lr = 0.03$, the model incurs a proportionally large classification error. Moreover, a popular technique in recent works for image classification is to use adaptive learning rate decay schedules that can encourage fast convergence [28]. Thus, we try two different learning rate schedules: stagewise step decay schedule [12] and cosine learning rate decay [18]. However, we do not obtain performance gain by using learning rate decay schedules under our SemiFed framework. Thus, we maintain the constant learning rate in all our experiments.

Table 3: Parameter study on learning rate and decay schedules. Accuracy is reported on CIFAR-10 with 4000 labeled samples and non-IID data

Decay Schedule	Validation Acc (%)
constant $lr = 0.3$	75.51
constant $lr = 0.03$	69.54
stagewise decay	73.14
cosine decay	73.63

4.4 Ablation Study In order to understand the influence of consistency loss and pseudo-labeling method individually in model performance, we perform an ablation study by isolating each procedure's effects.

1) *Benefit of applying consistency regularization.* In this section, we are interested in evaluating how consistency regularization can benefit exploiting information from unlabeled data. To isolate the contribution of consistency regularization in the neural network, we train the model using only the labeled data. Then, we apply the pseudo-labeling method to generate labels for unlabeled data to guide the learning process. Specifically, we compare it with the approach that only uses supervised data and our SemiFed algorithm. Figure 4 presents the results on CIFAR-10 under IID and non-IID settings.

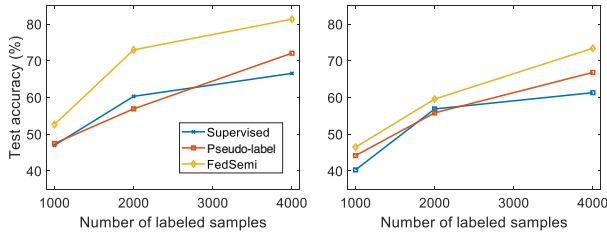


Figure 4: Test accuracy of using Supervised, Pseudo-labeling method without consistency loss and SemiFed algorithm on CIFAR-10. The left image shows the test accuracy under IID data partitions and the right image shows the test accuracy under non-IID data partitions

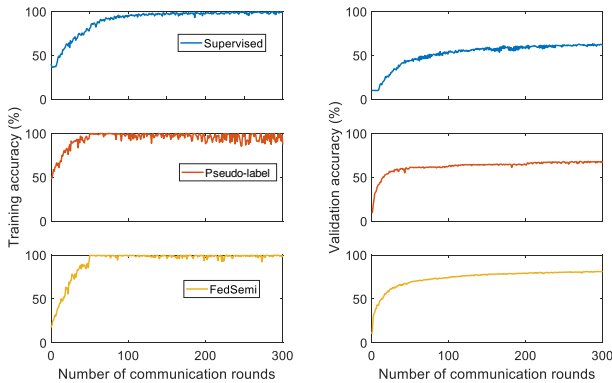


Figure 5: Training accuracy and validation accuracy of using Supervised, Pseudo-labeling method without consistency loss and SemiFed algorithm on CIFAR-10 with 4000 labeled images under non-IID data partitions

It is clear that consistency regularization plays a significant role in our model performance. To further understand the role of consistency regularization in network training, we present the validation accuracy and training accuracy (the accuracy is aggregated among all the clients) for CIFAR-10 with 4000 labeled samples in non-IID partitions (See Figure 5). From the curve we

can make the following observations: compared with the method that applies pseudo-labeling without consistency loss, our algorithm has a smoother curve in training accuracy. This is because the pseudo-labeling method introduces noisy labels to the dataset which may hurt model performance. Meanwhile, the validation accuracy of the supervised method that only trains with 4000 labeled samples also shows that the trained model does not have good generalization accuracy. The consistency regularization helps the network to fit the training data perfectly and provide better generalization performance [29, 30].

2) *Benefit of pseudo-labeling.* We are interested in evaluating how the pseudo-labeling benefits learning performance in SemiFed. Since UDA uses the same augmentation as SemiFed without pseudo-labeling, we compare federated UDA baselines against our SemiFed algorithm. We note that our method performs the best on all almost all experiments, verifying the benefit of pseudo-labeling. The only exception is for SVHN, where UDA is a bit superior with 4000 labeled data samples. This may be explained by the fact that image classification on SVHN is not a difficult task, and thus, we can get the trained model with relatively high accuracy on both the test set and the training set. As shown in the Table 2, the gap between the model trained on the fully supervised training samples and the SVHN with 4000 labeled data is small, and thus pseudo-labeling method cannot bring any performance gain to the network. In addition, we can also observe from Figure 4 that pseudo-labeling alone performs better than supervised approach in most cases, demonstrating the effectiveness of pseudo-labeling in the training procedure.

5 Conclusion

We proposed SemiFed, a unified framework to address the challenge in limited samples for image classification based on a combination of consistency regularization and pseudo-labeling. We adapt pseudo-labeling method to FL setting by allowing multiple models to agree on the same dataset to improve robustness accuracy of the network. The experimental results confirm the feasibility of our conceptually simple approach compared with baselines in two datasets under both IID and non-IID settings.

In the future, we plan to explore how to enhance synergies of pseudo-labeling and consistency regularization and design an automatic scheme for deciding the threshold or criteria used for adding training samples across communication rounds.

References

- [1] M. ABADI, A. AGARWAL, P. BARHAM, E. BREVDO, Z. CHEN, C. CITRO, G. S. CORRADO, A. DAVIS, J. DEAN, M. DEVIN, ET AL., *Tensorflow: Large-scale machine learning on heterogeneous distributed systems*, arXiv preprint arXiv:1603.04467, (2016).
- [2] P. BACHMAN, O. ALSHARIF, AND D. PRECUP, *Learning with pseudo-ensembles*, in NIPS, 2014, pp. 3365–3373.
- [3] D. BERTHELOT, N. CARLINI, E. D. CUBUK, A. KURAKIN, K. SOHN, H. ZHANG, AND C. RAFFEL, *Remixmatch: Semi-supervised learning with distribution matching and augmentation anchoring*, in ICLR, 2020.
- [4] D. BERTHELOT, N. CARLINI, I. GOODFELLOW, N. PAPERNOT, A. OLIVER, AND C. A. RAFFEL, *Mixmatch: A holistic approach to semi-supervised learning*, in NIPS, 2019, pp. 5049–5059.
- [5] S. CALDAS, P. WU, T. LI, J. KONECNY, H. B. MCMAHAN, V. SMITH, AND A. TALWALKAR, *Leaf: A benchmark for federated settings*, arXiv preprint arXiv:1812.01097, (2018).
- [6] M. CARON, P. BOJANOWSKI, A. JOULIN, AND M. DOUZE, *Deep clustering for unsupervised learning of visual features*, in ECCV, 2018, pp. 132–149.
- [7] E. D. CUBUK, B. ZOPH, D. MANE, V. VASUDEVAN, AND Q. V. LE, *Autoaugment: Learning augmentation policies from data*, arXiv preprint arXiv:1805.09501, (2018).
- [8] E. D. CUBUK, B. ZOPH, J. SHLENS, AND Q. V. LE, *Randaugment: Practical automated data augmentation with a reduced search space*, in CVPR Workshops, 2020, pp. 702–703.
- [9] I. J. GOODFELLOW, J. SHLENS, AND C. SZEGEDY, *Explaining and harnessing adversarial examples*, arXiv preprint arXiv:1412.6572, (2014).
- [10] N. GUHA, A. TALWALKAR, AND V. SMITH, *One-shot federated learning*, arXiv preprint arXiv:1902.11175, (2019).
- [11] C. HE, S. LI, J. SO, M. ZHANG, H. WANG, X. WANG, P. VEPAKOMMA, A. SINGH, H. QIU, L. SHEN, P. ZHAO, Y. KANG, Y. LIU, R. RASKAR, Q. YANG, M. ANNAVARAM, AND S. AVESTIMEHR, *Fedml: A research library and benchmark for federated machine learning*, arXiv preprint arXiv:2007.13518, (2020).
- [12] K. HE, X. ZHANG, S. REN, AND J. SUN, *Deep residual learning for image recognition*, in CVPR, 2016, pp. 770–778.
- [13] K. HSIEH, A. PHANISHAYEE, O. MUTLU, AND P. B. GIBBONS, *The non-iid data quagmire of decentralized machine learning*, arXiv preprint arXiv:1910.00189, (2019).
- [14] W. JEONG, J. YOON, E. YANG, AND S. J. HWANG, *Federated semi-supervised learning with inter-client consistency*, arXiv preprint arXiv:2006.12097, (2020).
- [15] S. LAINE AND T. AILA, *Temporal ensembling for semi-supervised learning*, arXiv preprint arXiv:1610.02242, (2016).
- [16] D.-H. LEE, *Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks*, in Workshop on challenges in representation learning, ICML, vol. 3, 2013.
- [17] F. LOCATELLO, S. BAUER, M. LUCIC, G. RAETSCH, S. GELLY, B. SCHÖLKOPF, AND O. BACHEM, *Challenging common assumptions in the unsupervised learning of disentangled representations*, in ICML, 2019, pp. 4114–4124.
- [18] I. LOSCHILOV AND F. HUTTER, *Sgdr: Stochastic gradient descent with warm restarts*, arXiv preprint arXiv:1608.03983, (2016).
- [19] D. MAHAJAN, R. GIRSHICK, V. RAMANATHAN, K. HE, M. PALURI, Y. LI, A. BHARAMBE, AND L. VAN DER MAATEN, *Exploring the limits of weakly supervised pretraining*, in ECCV, 2018, pp. 181–196.
- [20] H. MCMAHAN, E. MOORE, D. RAMAGE, S. HAMPSON, AND B. A. Y. ARCAS, *Communication-efficient learning of deep networks from decentralized data*, in AISTATS, 2017.
- [21] T. MIYATO, S.-I. MAEDA, M. KOYAMA, AND S. ISHII, *Virtual adversarial training: a regularization method for supervised and semi-supervised learning*, TPAMI, 41 (2018), pp. 1979–1993.
- [22] Y. NETZER, T. WANG, A. COATES, A. BISSACCO, B. WU, AND A. Y. NG, *Reading digits in natural images with unsupervised feature learning*, (2011).
- [23] U. OF TORONTO, *Learning multiple layers of features from tiny images*, (2012).
- [24] A. OLIVER, A. ODENA, C. A. RAFFEL, E. D. CUBUK, AND I. GOODFELLOW, *Realistic evaluation of deep semi-supervised learning algorithms*, in NIPS, 2018, pp. 3235–3246.
- [25] A. PASZKE, S. GROSS, F. MASSA, A. LERER, J. BRADBURY, G. CHANAN, T. KILLEEN, Z. LIN, N. GIMELSHEIN, L. ANTIGA, ET AL., *Pytorch: An imperative style, high-performance deep learning library*, in NIPS, 2019, pp. 8026–8037.
- [26] T. RYFFEL, A. TRASK, M. DAHL, B. WAGNER, J. MANCUSO, D. RUECKERT, AND J. P. PALMBACH, *A generic framework for privacy preserving deep learning*, CoRR, abs/1811.04017 (2018).
- [27] W. SHI, Y. GONG, C. DING, Z. MAXIAOYU TAO, AND N. ZHENG, *Transductive semi-supervised deep learning using min-max features*, in ECCV, 2018, pp. 299–315.
- [28] K. SOHN, D. BERTHELOT, C.-L. LI, Z. ZHANG, N. CARLINI, E. D. CUBUK, A. KURAKIN, H. ZHANG, AND C. RAFFEL, *Fixmatch: Simplifying semi-supervised learning with consistency and confidence*, arXiv preprint arXiv:2001.07685, (2020).
- [29] A. TARVAINEN AND H. VALPOLA, *Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results*, in NIPS, 2017, pp. 1195–1204.
- [30] Q. XIE, Z. DAI, E. HOVY, M.-T. LUONG, AND Q. V. LE, *Unsupervised data augmentation for consistency training*, arXiv preprint arXiv:1904.12848, (2019).
- [31] M. YUROCHKIN, M. AGARWAL, S. GHOSH, K. GREE-NEWALD, T. N. HOANG, AND Y. KHAZAENI, *Bayesian*

nonparametric federated learning of neural networks,
arXiv preprint arXiv:1905.12022, (2019).

- [32] X. ZHU, *Semi-supervised learning literature survey*,
(2005).